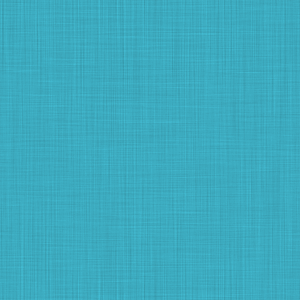
**Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**



**CS**101

lab03



**Objectives**

* Use indexing to access components of a data structure.
* Explain blocks and variable scope and understand how it impacts variable use.

**Modular Code (Blocks and Scope)**

Effective programs are broken up into a natural hierarchy of

operations. The resulting blocks each have a specific task and are executed as a unit. Consider this program to calculate the pressure of an ideal gas given the temperature and volume:

*P* ∝ *T*/*V* = *RT*/*V*

|  |  |
| --- | --- |
| 0  1  2  3  4  5  6  7  8  9  10  11 | def pressure\_IG(T, V):  # T should be in deg C and V in cubic meters  R = 8.314 # ideal gas constant, joules / deg C  P = R \* T / V  return P  temperature = 100.0 # deg C  volume = 0.01 # cubic meters  pressure = pressure\_IG(temperature, volume)  print('The pressure of', volume, 'cubic meters of gas at', temperature,  'deg C is', pressure, 'pascals') |

This program consists of two blocks: the outermost layer (lines 0, 5–11) and the function body (lines 1–4). Python executes this program as follows:

* 1. Line 0—Python notices that we have created a function pressure\_IG that accepts two arguments.
  2. Lines 6–7—Python creates two variables, temperature and volume.
  3. Line 8—Python attempts to create a variable named pressure. But in order to do so, Python finds it needs to look at the block of code referred to by the function pressure\_IG. So Python takes the *values* of temperature and volume and places them in T and V.
  4. Lines 1–4—Python calculates the value of P and returns it to the calling code location.
  5. Line 8—Python completes the creation of the variable pressure with value equal to the returned value of pressure\_IG.
  6. Line 10—Python outputs the results in temperature, volume, and pressure.

1. Draw arrows and labels in the code above to describe the control flow (following the text).

Consider another program, this one defining a function to square input numbers:

|  |  |
| --- | --- |
| 0  1  2  3  4  5 | def sqr( x ):  return x \*\* 2  x = 5  y = 3  print( sqr( y ) ) |

Here we see *two* variables x—one defined inside of the function sqr and the other in the main block of code. We use the concept of *variable scope* to understand what each x means where. Basically, if we have a single block of code, then any reference to a variable or name (such as x) is interpreted by Python to mean the *local* x, or the x within that block.

1. What is the value of the current (in-scope) x *after* line 3 executes? \_\_\_\_\_\_\_\_
2. What is the value of the current (in-scope) x *after* line 1 executes (given the call on line 5)?

\_\_\_\_\_\_\_\_

1. What is the value of the current (in-scope) x *after* line 5 executes? \_\_\_\_\_\_\_\_